FLEXIBLE ASSEMBLY SYSTEMS

Job Shop and Flexible Assembly

Job Shop
- Each job has an unique identity
- Make to order, low volume environment
- Possibly complicated route through system
- Very difficult

- **Flexible Assembly**
- Limited number of product types
- Given quantity of each type
- Mass production
- High degree of automation
- **Even more difficult!**

Flexible Assembly Systems

- Sequencing Unpaced Assembly Systems
  - Simple flow line with finite buffers
  - ➢ Application: assembly of copiers
- Sequencing Paced Assembly Systems
  - Conveyor belt moves at a fixed speed
  - ➢ Application: automobile assembly
- Scheduling Flexible Flow Systems
  - Flow lines with finite buffers and bypass
  - ➢ Application: producing printed circuit board

Objectives

- Multiple objectives usual
  - Meet due dates

$$\sum w_j T_j$$

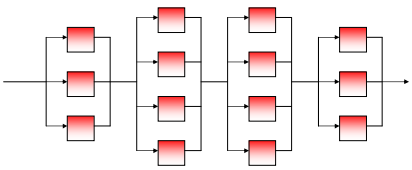
  - Maximize throughput

$$\sum s_{ijk} = \sum h_i(a_{ik}, a_{ij})$$

  - Minimize work-in-process (WIP)

Setting for job *j* on machine *i*

Flexible Flow Shop



Unpaced Assembly Systems

- Unpaced: machines can spend as much time as needed on any job
- Machines (workstations) in series
- Material handling system
  - When a job finishes moves to next station
  - No bypassing (First In First Out)
  - Blocks job if buffer is full
- Can model any *finite buffer* situation. A buffer is modeled as a "machine" with zero processing time.

Cyclic schedules

- Schedules often cyclic or periodic
  - Given set of jobs scheduled in certain order
    - Contains all product types
    - May contain multiple jobs of same type
  - Second identical set scheduled, etc.
- Practical if insignificant setup time
  - Low inventory holding costs
  - Easy to implement
- Cyclic schedules may **not** optimize throughput!

Minimum Part Set

- Suppose there are l different product types
- Let N_k be number of jobs of product type k
- Let z be the greatest common divisor of $N_1,\ldots, N_l$
- Then $N^* = \left(\frac{N_1}{z}, \frac{N_2}{z}, \ldots, \frac{N_l}{z}\right)$

  is the smallest set with same proportions as the long range production set.
- It is called the **Minimum Part Set** (MPS)

Defining a cyclic schedule

- Consider the jobs in the MPS as n jobs

$$n = \frac{1}{z}\sum_{k=1}^{l} N_k = \sum_{k=1}^{l} N_k^*$$

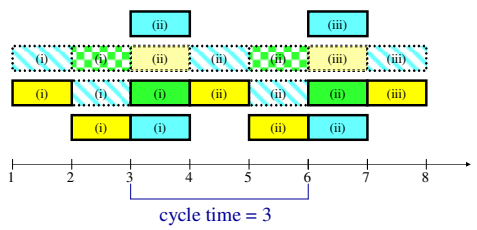
- Let p_{ij} be the processing time as before
- A cyclic schedule is determined by sequencing jobs in the MPS
- **Maximizing throughput** is equivalent to **minimizing cycle time** in a steady-state.

MPS Cycle Time Example

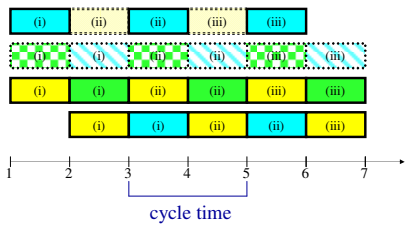
Jobs	1	2	3
p_{1j}	0	1	0
p_{2j}	0	0	0
p_{3j}	1	0	1
p_{4j}	1	1	0

buffer

Sequence: 1, 2, 3



Sequence: 1, 3, 2



Minimizing cycle time

- **Profile Fitting** (PF) heuristic:
  - Select first job j_1
    - Arbitrarily
    - Largest amount of processing
  - Generates profile. Departure of job 1 from machine ***i***:

$$X_{i,j_1} = \sum_{h=1}^{i} p_{h,j_1}$$

  - Determine which job goes next

Profile Fitting: next job

- Compute for each candidate job
  - time the machines are idle
  - time the job is blocked
  - departure times of a job c is computed recursively by:

$$X_{1,j_2} = \max\left(X_{1,j_1} + p_{1c}, X_{2,j_1}\right)$$
$$X_{i,j_2} = \max\left(X_{i-1,j_2} + p_{ic}, X_{i+1,j_1}\right), \quad i = 2,\ldots,m-1$$
$$X_{m,j_2} = X_{m-1,j_2} + p_{mc}$$

Nonproductive time

- Calculate sum of idle and blocked time for job c

$$\sum_{i=1}^{m}\left(X_{i,j_2} - X_{i,j_1} - p_{ic}\right)$$

- Repeat for all remaining jobs in the MPS
- Select job with smallest number

- Calculate new profile and repeat

Profile Fitting heuristic

Step 1: *Initial Condition*

- *Select the job with the longest total processing time as the first job in the MPS.*

Step 2: *Analysis of remaining jobs to be scheduled*

- *For each not yet scheduled: consider it the next one in the partial sequence and compute the total non-productive time on all* m *machines (machine idle time and machine blocking time).*

Profile Fitting heuristic

Step 3: *Select the next job in partial schedule*

- *From all jobs analyzed in Step 2, select the job with the smallest total non-productive time as the next in the partial sequence.*

Step 4: *Stopping criterion*

- *If all jobs in the MPS have been scheduled, then STOP*
- *Otherwise, go to Step 2.*

Discussion: PF Heuristic

- PF heuristic performs well in practice
- Refinement:
  - Nonproductive time is not equally bad on all machines
  - Bottleneck machine are more important.
  - Use weight in the sum: **Weighted Profile Fitting heuristic** (see Example 6.2.3)

Discussion: PF Heuristic

- Basic assumptions
  - Setup time is not important
  - Low WIP is important
    - ⇒ Cyclic schedules are good
- Want to maximize throughput
  - ⇒ Minimize cycle time
  - ⇒ PF heuristic performs well

Discussion: solution methods

- Formulated as 'simple' sequencing
- Can apply branch-and-bound
- In general constraints make mathematical programming formulation difficult

- **PF heuristic is easy to generalize**

Additional complications

- The material handling system does not wait for a job to be complete
  - ⇒ **Paced assembly systems**

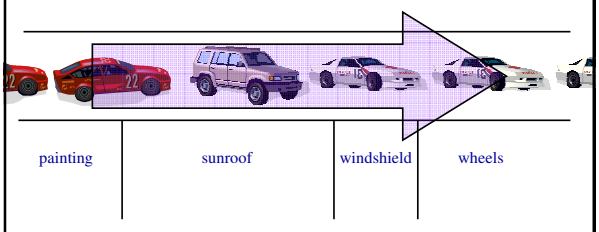
- There may be multiple machines at each station and/or there may be bypass
  - ⇒ **Flexible flow systems with bypass**

Paced Assembly Systems

- Conveyor moves jobs from one workstation to another at fixed speeds
- Fixed distance between jobs
  - Spacing proportional to processing time
- No bypass
- Unit cycle time
  - time between two successive jobs
  - line balancing

➢ **Example**: automotive industry

Car assembly lines



Paced assembly lines

- Line runs at a constant rate
- Size of work centre proportional to duration of operation
  - It takes longer to install a sunroof than a windshield so the workstation is longer
- Once the setup is done, the only thing to control is the sequence of cars

Car sequencing minimizing setup cost

- n cars
- m paint colors
- Color of each car is known
- Every time color is changed it has a cost (for cleaning, discarding paint, etc.)
  - Setup or transition cost

➢ **Problem**: Find a sequence that minimizes setup cost

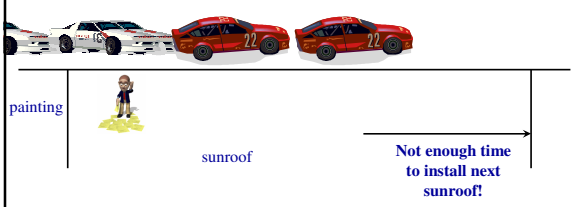
Car sequencing with options

Each kind of car requires a certain set of options

- Sunroof, air conditioning, spoiler, etc.
- The work centre for each option is capacity constrained

Car sequencing with options

- Can only install sunroofs in 1 out of every 3 cars



Car sequencing with options

- Can only install sunroofs in 1 out of every 3 cars
- But it gets worse
  - Every car requires a *set* of options
  - Each option has a capacity constraint

"Full" car sequencing problem

- Minimize total setup cost
- Minimize distance from pre-assigned slot
  - Make-to-order jobs
- Capacity constraints on options
- Make the rate of consumption of parts at each station as constant as possible

Objectives in paced assembly systems

- Minimize total setup cost
  - Ex: paint shop a job with color j is followed by color k has a setup cost of c_{jk}.
- Meet due dates for make-to-order jobs
  - Total weighted tardiness
- Spacing of capacity constrained operations
  - $\psi_i(\ell)$ = penalty for working on two jobs ℓ positions apart in i^{th} workstation. Decreases with ℓ (as installing sunroofs)
- Keep rate of material consumption as regular as possible at all workstations.

Possible algorithms

- Extend Constrained Programming model
  - Setup costs – easy to add
  - Distance from preassigned slots – easy to add
  - See Optimization Programming Language code in Appendix D.4 of Pinedo's book
- Dispatch rules, IP formulation, Tabu search
- Ant Colony Optimization
- ➢ **Heuristics** – **Grouping and Spacing** (GS)

Grouping and Spacing heuristic

i. Determine the total number of jobs to be scheduled
ii. Group jobs with high setup cost operations
iii. Order each subgroup accounting for shipping dates
iv. Space jobs within subgroups accounting for capacity constrained operations

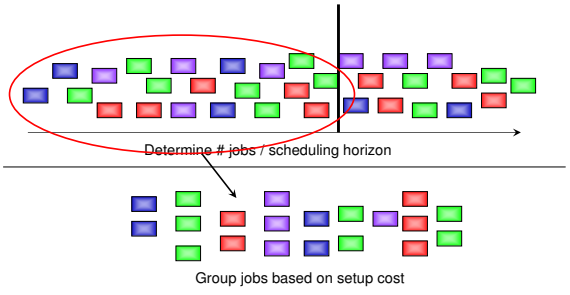
Grouping and Spacing heuristic

- Total number of jobs
  - High number allows a sequence with lower cost. However, probability of disruption is high.
  - Number of jobs is normally between one day and one week.
- Grouping jobs
  - Ex: determining run lengths of different colors. Grey can go up to 50, and purple may be only 1 or 2.
  - Other examples: options, destination of cars.
  - Similar to computing an Economic Order Quantity (see Lot Scheduling)

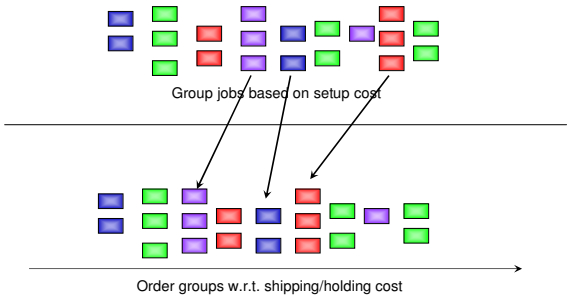
Grouping and Spacing heuristic

- Ordering different subgroups
  - Determined by the urgency with which the jobs in a group have to be shipped.
  - Urgency is determined by *committed shipping dates* of Make–To–Order jobs and *current inventory levels* of Make–To–Stock jobs.
- Internal sequencing within subgroups
  - Consider *capacity constrained* operations.
  - Consider most critical operation and space jobs as uniformly as possible. It proceeds similarly to the other operations.

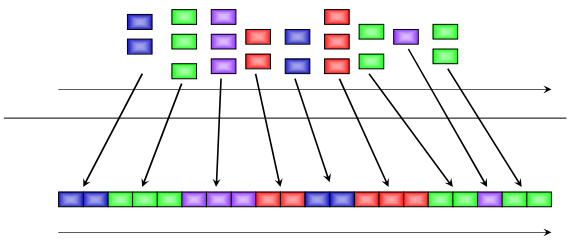
Grouping and Spacing heuristic



Grouping and Spacing heuristic



Grouping and Spacing heuristic



Sequence jobs within groups to account capacity constraints

Example 6.3.2

- Single machine, 10 jobs with two attributes. $p_j = 1, \forall j$
- a_{j1}: color; a_{j2}=1: with sunroof.
- If $a_{j1} \neq a_{k1}$ the setup cost is: $c_{jk} = |a_{j1} - a_{k1}|$
- If $a_{j2} = a_{k2} = 1$ and spaced ℓ jobs apart the penalty cost is: $\psi_2(\ell) = \max(3 - \ell, 0)$
- Tardiness $w_j T_j$ is considered for job j with due date d_j.

➢ **Find sequence that minimize the total cost.**

Example data

Job	1	2	3	4	5	6	7	8	9	10
a_{1j}	1	1	1	3	3	3	5	5	5	5
a_{2j}	0	1	1	0	1	1	1	0	0	0
d_j	∞	2	∞	∞	∞	∞	6	∞	∞	∞
w_j	0	4	0	0	0	0	4	0	0	0

Grouping

- Group A: Jobs 1, 2 and 3
- Group B: Jobs 4, 5 and 6
- Group C: Jobs 7, 8, 9 and 10

➢ Best order according to setup cost: A → B → C

Grouped jobs

	A			B			C			
Job	1	2	3	4	5	6	7	8	9	10
a_{1j}	1	1	1	3	3	3	5	5	5	5
a_{2j}	0	1	1	0	1	1	1	0	0	0
d_j	∞	2	∞	∞	∞	∞	6	∞	∞	∞
w_j	0	4	0	0	0	0	4	0	0	0

Due date

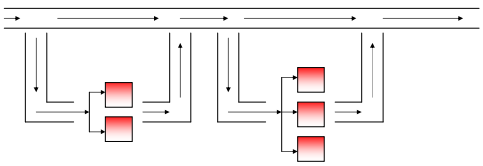
⇒ Order A → C → B

Capacity constrained operations

	A			C				B		
Job	2	1	3	8	7	9	10	5	4	6
a_{1j}	1	1	1	5	5	5	5	3	3	3
a_{2j}	1	0	1	0	1	0	0	1	0	1
d_j	2	∞	∞	∞	6	∞	∞	∞	∞	∞
w_j	4	0	0	0	4	0	0	0	0	0

Total cost: 6 (setup cost) + 3 (capacity constraint) = 9

Flexible Flow Systems



Flexible Flow Line Loading algorithm

- **Objectives**
  - Maximize throughput
  - Minimize Work-In-Process (WIP)
- Minimizes the makespan of a day's mix
  - Actually minimization of cycle time for Minimum Part Set (MPS)
- Reduces blocking probabilities

Flexible Flow Line Loading algorithm

Three phases:

i. Machine allocation phase
   - assigns each job to a specific machine at each stage

ii. Sequencing phase
   - orders in which jobs are released
   - Dynamic Balancing heuristic

iii. Time release phase
   - minimize MPS cycle time on bottlenecks

FFLL algorithm: machine allocation

- Bank of machines
- Which machine for which job?
- Basic idea: workload balancing
- ➢ **Use LPT dispatching rule**

- Output: allocation of jobs to machines, but not the sequencing of jobs or the timing of processing.

FFLL algorithm: sequencing

- Basic idea: spread out jobs sent to the same machine
- *Dynamic Balancing* heuristic
- For a given station, let p_{ij} be processing time of job j on machine i.
- Let $W_i = \sum_{j=1}^{n} p_{ij}$
- and the total workload of an MPS be

$$W = \sum_{i=1}^{m} W_i$$

Dynamic Balancing heuristic

- Let J_k be set of jobs loaded into the system, including job k
- Total workload of machine i that entered the system by the time job k is loaded:

$$\alpha_{ik} = \sum_{j \in J_k} \frac{p_{ij}}{W_i} \in [0,1]$$

- Ideal balanced target:

$$\alpha_k^* = \sum_{j \in J_k} \sum_{i=1}^{m} p_{ij} \Big/ \sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij} = \sum_{j \in J_k} p_j / W$$

Minimizing overload

- *Overload* of machine i due to job k

$$o_{ik} = p_{ik} - p_k W_i / W$$

- Cumulative overload of all jobs up to and including job k

$$O_{ik} = \sum_{j \in J_k} o_{ij} = \sum_{j \in J_k} p_{ij} - \alpha_k^* W_i$$

➢ **Objective**: minimize

$$\sum_{k=1}^{n} \sum_{j=1}^{m} \max(O_{ik}, 0)$$

FFLL algorithm: release timing

- MPS workload of each machine known
  - Highest workload is the bottleneck
  - MPS cycle time ≥ Bottleneck cycle time

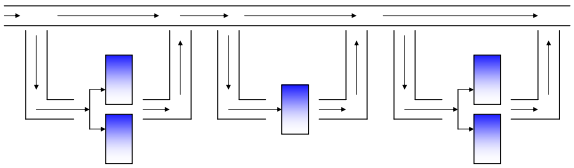
Algorithm

➢ **Step 1**: Release all jobs as soon as possible

➢ **Step 2**: Delay all jobs upstream from bottleneck as much as possible

➢ **Step 3**: Move up all jobs downstream from the bottleneck as much as possible

Example 6.4.1



Data

p'_{hj} : processing time of job j at stage h

Jobs	1	2	3	4	5
p'_{1j}	6	3	1	3	5
p'_{2j}	3	2	1	3	2
p'_{3j}	4	5	6	3	4

Machine allocation

- Using LPT:

Jobs	1	2	3	4	5
p_{1j}	6	0	0	3	0
p_{2j}	0	3	1	0	5
p_{3j}	3	2	1	3	2
p_{4j}	4	5	0	3	0
p_{5j}	0	0	6	0	4

6 3
5 3 1
5 4 3
6 4

Workload

- From the table we obtain

each row	each column
$W_1 = 9$	$p_1 = 13$
$W_2 = 9$	$p_2 = 10$
$W_3 = 11$	$p_3 = 8$
$W_4 = 12$	$p_4 = 9$
$W_5 = 10$	$p_5 = 11$
$W = 51$	

Overload

- With job 1 as first job of sequence:

$$o_{11} = 6 - 9 \times 13/51 = +3.71$$
$$o_{21} = 0 - 9 \times 13/51 = -2.29$$
$$o_{31} = 3 - 11 \times 13/51 = +0.20$$
$$o_{41} = 4 - 12 \times 13/51 = +0.94$$
$$o_{51} = 0 - 10 \times 13/51 = -2.55$$

Overload matrix

- With jobs 1 to 5 as first job of sequence, we can compute matrix o_{ik}

3,71	-1,76	-1,41	1,41	-1,94
-2,29	1,24	-0,41	-1,59	3,06
0,20	-0,16	-0,73	1,06	-0,37
0,94	2,65	-1,88	0,88	-2,59
-2,55	-1,96	4,43	-1,76	1,84

Dynamic Balancing heuristic

3,71	0,00	0,00	1,41	0,00
0,00	1,24	0,00	0,00	3,06
0,20	0,00	0,00	1,06	0,00
0,94	2,65	0,00	0,88	0,00
0,00	0,00	4,43	0,00	1,84
4,84	3,88	4,43	**3,35**	4,90

First Job

Selecting the second job

- Calculate the cumulative overload

$$O_{11} = \sum_{j \in J_1} p_{1j} - \alpha_1^* W_1$$
$$= \sum_{j \in \{4,1\}} p_{1j} - 0.43 \times 9$$
$$= (3+6) - 0.43 \times 9$$
$$= 5.12$$

where $\alpha_1^* = \sum_{j \in J_k} p_j / W = \sum_{j \in \{4,1\}} p_j / 51$

$$= (9+13)/51 = 0.43$$

Cumulative overload

$$O_{i1} = (+5.11, -3.88, +1.26, +1.82, -4.32)$$
$$O_{i2} = (-0.35, -0.36, +0.90, +3.52, -3.72)$$
$$O_{i3} = (+0.00, -2.00, +0.33, -1.00, +2.67)$$
$$O_{i5} = (-0.53, +1.47, +0.69, -1.71, +0.08)$$

Selected next

Final cycle

- Schedule jobs 4, 5, 1, 3, 2

- Release timing phase
  - Machine 4 is the bottleneck
  - Delay jobs on Machine 1, 2, and 3
  - Expedite jobs on Machine 5

Assembly sequence at Toyota

- Everything operates on Just-in-Time
  - Works to minimize WIP and tardiness
- Most important objective is to keep the part consumption regular
  - The quantity of a given part consumed per hour should be constant
- ➢ **Solution**: the next car to build is chosen to minimize error from the desired rate

Example

- Model 1 requires 1 *unit*
- Model 2 requires 2 *units*
- Model 3 requires 3 *units*
- Production needed:
  - 10 cars of model 1
  - 15 cars of model 2
  - 10 cars of model 3

Example

- So, it is needed
  - $(10 \times 1) + (15 \times 2) + (10 \times 3) = 70$
- If the consumption of *unit* is to be constant, the consumption is 70/35 = 2 per car
- Choose car *j* to minimize:
  - (*units* use / cars built – 2)^2

Toyota problem

- A car has around 20000 number of parts.
- Parts are represented by their respective subassembly
- Number of subassemblies in Toyota is around 20.
- Toyota developed the Goal Chasing Method to solve the mixed model assembly, which minimizes the sum of squared error over all subassemblies.
- See p.134

Flexible Manufacturing Systems

- **Flexible Manufacturing Systems** (FMS)
  - Numerically Controlled machines
  - Automated Material Handling system
  - Produces a variety of product/part types
- Scheduling
  - Routing of jobs
  - Sequencing on machines
  - Setup of tools
- Similar features but more complicated